Pattern Recognition 127 (2022) 108643

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

BeCAPTCHA-Mouse: Synthetic mouse trajectories and improved bot detection

Alejandro Acien*, Aythami Morales*, Julian Fierrez, Ruben Vera-Rodriguez

Biometrics and Data Pattern Analytics Lab, Universidad Autonoma de Madrid, Spain

ARTICLE INFO

Article history:
Received 26 February 2021
Revised 30 November 2021
Accepted 8 March 2022
Available online 14 March 2022

Keywords:
CAPTCHA
Bot detection
Behavior
Biometrics
Mouse
Neuromotor

ABSTRACT

We first study the suitability of behavioral biometrics to distinguish between computers and humans, commonly named as bot detection. We then present BeCAPTCHA-Mouse, a bot detector based on: *i)* a neuromotor model of mouse dynamics to obtain a novel feature set for the classification of human and bot samples; and *ii)* a learning framework involving real and synthetically generated mouse trajectories. We propose two new mouse trajectory synthesis methods for generating realistic data: *a)* a function-based method based on heuristic functions, and *b)* a data-driven method based on Generative Adversarial Networks (GANs) in which a Generator synthesizes human-like trajectories from a Gaussian noise input. Experiments are conducted on a new testbed also introduced here and available in GitHub: BeCAPTCHA-Mouse Benchmark; useful for research in bot detection and other mouse-based HCI applications. Our benchmark data consists of 15,000 mouse trajectories including real data from 58 users and bot data with various levels of realism. Our experiments show that BeCAPTCHA-Mouse is able to detect bot trajectories of high realism with 93% of accuracy in average using only one mouse trajectory. When our approach is fused with state-of-the-art mouse dynamic features, the bot detection accuracy increases relatively by more than 36%, proving that mouse-based bot detection is a fast, easy, and reliable tool to complement traditional CAPTCHA systems.

1. Introduction

During the last decades, the security applications have had a key role in the development of machine learning technologies. Thus, research areas such as fingerprint identification, face recognition, iris recognition, or person re-identification have attracted the interest of the research community promoting continuous advances in their fields. These advances resulted in more accurate physical security systems and advances in state-of-the-art. However, security threats are moving from the physical domain to the digital domain. The Cybercrime is increasing in both percentage of citizens affected and cost in the global economy[1]. The criminals become more and more sophisticated and their crimes have a cross-border scope. The challenges and potential benefits of technologies developed to serve in this fight are large and the Pattern Recognition community can play an important role in this scenario. Among these challenges, the present work is focused on the detection of bots and how pattern recognition techniques and machine learning frameworks can be used to develop new approaches.

How to distinguish between human users and artificial intelligence during computer interactions is not a trivial task. This challenge was firstly discussed by Alan Turing in 1950. He investigated whether machines could show an intelligent behavior, and also how humans could be aware of these artificial behaviors. For this, he developed the famous Turing Test, commonly named as *The Imitation Game*, in which a human evaluator would judge natural language conversations between a human and a computer designed to generate human-like responses. The Turing Test was both influential and widely criticized and became an important concept in the artificial intelligence field [1]. However, at the epoch of Alan Turing research, the problem of machines acting like humans were commonly associated to science-fiction topics.

Nowadays, boosted by the last advances of machine learning technologies and worldwide connections, that 'science-fiction topic' becomes a real hazard. As an example, bots are expected to be responsible for more than 40% of the web traffic with more than 43% of all login attempts to come from malicious botnets in the next years[2]. Malicious bots cause billionaire losses through web

* Corresponding author.
E-mail addresses: alejandro.acien@uam.es (A. Acien), aythami.morales@uam.es (A. Morales), julian.fierrez@uam.es (J. Fierrez), ruben.vera@uam.es (R. Vera-Rodriguez).

[1] https://www.cbronline.com/news/cybercrime-cost-fbi

[2] https://resources.distilnetworks.com/white-paper-reports/bad-bot-report-2019

https://doi.org/10.1016/j.patcog.2022.108643
0031-3203/

scraping, account takeover, account creation, credit card fraud, denial of service attacks, denial of inventory, and many others. Moreover, bots are used to influence and divide society (e.g. usage of bots to interfere during Brexit voting day [2], or to spread anxiety and sadness during the COVID-19 outbreak[3,4] through Twitter). Bots are becoming more and more sophisticated, being able to mimic human online behaviors. On the other hand, algorithms to distinguish between humans and bots are also getting very complex. We can distinguish two types of bot detection methods in response to those sophisticated bots:

- Active Detection. Traditionally named as CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), these algorithms determinate whether or not the user is human by performing online tasks that are difficult for software bots to solve while being easy for legitimate human users to complete. Some of the most popular CAPTCHA systems are based on: characters recognition from distorted images (text-based), class-objects identification in a set of images (image-based), and speech translation from distorted audios (audio-based).
- Passive Detection. These detectors are transparent and analyze the users behavior while they interact with the device. The last version of Google reCAPTCHA v3 replaces traditional cognitive tasks by a transparent algorithm capable of detecting bots and humans from their web behavior[5]. Other researchers [3], describe browsing behavior of web users for detection of DDoS Attacks (Distributed Denial of Service).

Although these algorithms are broadly used, they present limitations. First of all, ensuring a very accurate bot detection makes the tasks difficult to perform even for humans. Second, most of the CAPTCHA systems can be easily solved by the most modern machine learning techniques. For example, the text-based CAPTCHA was defeated by Bursztein et al. [4] with 98% accuracy using a ML-based system to segment and recognize the text. In [5], the authors designed an AI-based system called unCAPTCHA to break Google's most challenging audio reCAPTCHAs. The last version of the Google CAPTCHA, named reCAPTCHAv3, was systematically fooled in [6] by synthesizing mouse trajectories using reinforcement learning techniques. Third, these algorithms process sensitive information and there are important concerns about how they comply with new regulations such as the European GDPR[6]. Fourth, the CAPTCHA systems become a great barrier to people with visual or other impairments. Finally, the Turing Test was designed as a task in which machines had to prove they were human, meanwhile in current CAPTCHA systems humans have to prove they are not machines (e.g. *Iam not a robot* from Google's). This means that the responsibility to prove the user's 'humanity' falls over human users instead of bots. At this point, there is still a large room for improvement towards reliable bot detection able to stop malicious software not bothering human users during natural web browsing.

On the other hand, Machine Learning and Pattern Recognition communities have made great advances during the last decades. These advances have boosted several research fields including Computer Vision, Audio Processing, and Natural Language Processing. Nonetheless, the application of these advances to the bot detection field has been rather low. While previous works [4,5] focus their efforts in beating the existing CAPTCHA systems and exposing their vulnerabilities with the latest advances in machine learning techniques, we use them to develop better bot detectors and harden the existing ones.

Biometric recognition refers to the automated recognition of individuals based on their physiological (e.g. fingerprint, face) and behavioral (e.g. keystroke, gait) characteristics [7–9]. Utilizing behavioral-based biometrics for improving the security against bots and other kind of attacks has been only studied very timidly [10]. Some examples in this regard using behavioral features to train cognitive models to parameterize the user behavior and detect patterns useful to improve the security of digital services can be found in [11–13].

Behavioral biometrics have been applied successfully in bot detection for mobile devices scenarios [14]. The method proposed in [14] combines information from the accelerometer and touchscreen sensors. However, in that work the software-based sampling rate of mobile devices and the simplicity of touch over touchscreens limited the results. Here we apply similar ideas to [14] considering in this case mouse dynamics instead of touchscreen gestures, a richer signal in terms of time resolution, naturalness, and neuromotor information [15].

To overcome these limitations, our contributions with this work go a step forward in the bot detection field for mouse dynamics, incorporating behavioral modeling and improved learning methods based on realistic synthetic samples (see Fig. 1). The two main contributions of this work are:

- (1) We propose BeCAPTCHA-Mouse, a new bot detector based on neuromotor features [15] obtained from kinematics models of the mouse trajectories. BeCAPTCHA-Mouse is trained with human and synthetic data generated to mimic human-being characteristics.
- (2) In order to train and evaluate BeCAPTCHA-Mouse and future approaches, we propose two new methods for generating realistic mouse trajectories: *i)* a Function-based method based on heuristic functions, and *ii)* a data-driven method based on GANs in which a Generator synthesizes human-like trajectories from a Gaussian noise input. We demonstrate the usefulness of these synthetic trajectories to train more accurate bot detectors. These Generators can be helpful in many HCI research areas and applications.

These main contributions are supported by an extensive experimentation developed on the basis of reproducible frameworks and a new publicly available dataset. The secondary contributions of this work can be summarized as:

- (3) Our experiments consider a large number of state-of-the-art classifiers and provide a detailed study, exposing the strengths and weakness of the classifiers in different scenarios. The experiments include: Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), and deep learning architectures such us Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs). These algorithms are evaluated for mouse trajectories with different characteristics (e.g. direction, length) and learning strategies (e.g. number of samples, supervised, non supervised).
- (4) We present BeCAPTCHA-Mouse Benchmark[7], the first public benchmark for mouse-based bot detection including 10,000 human and synthetic trajectories generated according to 10 different types of synthesized behaviors. The inclusion of various types of synthetic samples (both for training and testing BeCAPTCHA-Mouse) allows to train strong bot detectors. Also, it allows comprehensive evaluations under various conditions including the worst-case scenario in which bot attacks mimic human behavior using latest machine learning advances. This

[3] www.washingtonpost.com/science/2020/03/17/analysis-millions-coronavirus-tweets-shows-whole-world-is-sad/

[4] https://www.sciencealert.com/bots-are-causing-anxiety-by-spreading-coronavirus-misinformation

[5] https://www.google.com/recaptcha/intro/v3.html

[6] https://complianz.io/google-recaptcha-and-the-gdpr-a-possible-conflict/

[7] https://github.com/BiDAlab/BeCAPTCHA-Mouse

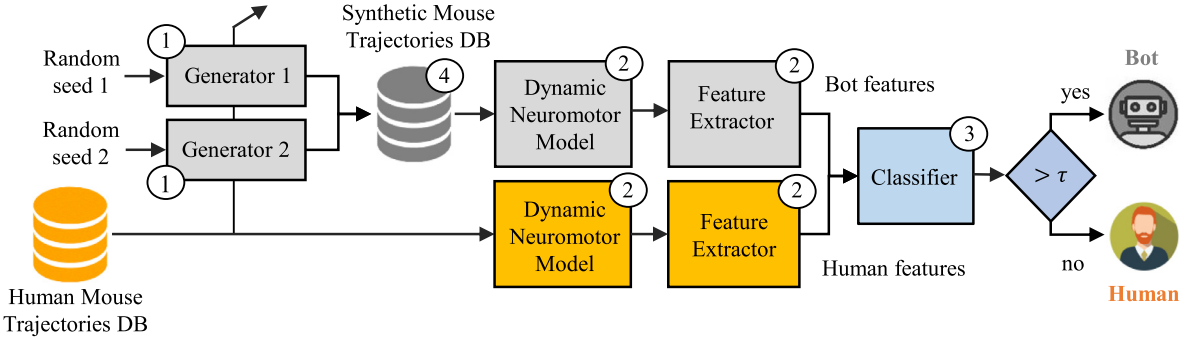


Fig. 1. Learning framework of BeCAPTCHA-Mouse: (1) We propose two novel methods to generate realistic synthetic mouse trajectories that allow to train and evaluate bot detection systems based on mouse dynamics; (2) We propose a neuromotor model to characterize Human and Synthetic Mouse Trajectories; (3) We evaluate the proposed features using multiple classifiers and learning scenarios; and (4) The proposed Generators can be also helpful for other HCI applications.

benchmark can be helpful for other HCI applications involving mouse dynamics beyond bot detection.

The main drawback of traditional CAPTCHA methods is that they only measure cognitive human skills (e.g. character recognition from distorted images, class-objects identification in a set of images, or speech translation from distorted audios). Trying to ensure a very accurate bot detection makes these CAPTCHAs difficult to perform even for humans. The main goal of our proposed method is to focus more on human behavioral skills rather than on cognitive ones. Neuromotor skills reveal human features useful for bot detection just with simple mouse trajectories. To the best of our knowledge, there are only a very limited number of works using mouse biometrics for bot detection. The most related to our research are [16] and [6]. In [6] they synthetize mouse trajectories over a grid to hack the Google reCAPTCHA v3 algorithm, and in [16] they extract global features (e.g. duration, average speed, displacement) from mouse and keystroke patterns to conduct a case study in the detection of blog bots for online blogging systems. While previous work in mouse dynamics ([16,17]) focused on basic cues like duration or average speed, in this work we go a step forward by focusing on the analysis and synthesis of entire mouse trajectories. We propose to use the Sigma-Lognormal model to extract human features that characterizes better human behaviors and novel generation methods to synthesize human-like trajectories to improve the training and evaluation of these methods. As shown in Fig. 2, our proposed mouse detection algorithm can be added in a transparent setup and enhance traditional CAPTCHAs based on cognitive challenges, for example when you select the images in a visual CAPTCHA, or when you navigate through a website.

The rest of the paper is organized as follows. In Section 2 we first discuss the usage of mouse dynamics in the context of behavioral biometrics. Section 3 presents our bot detector BeCAPTCHA-Mouse. Section 3.1 introduces the mouse dynamics neuromotor model and the features employed for the classification of bot and human trayectories. Section 3.2 describes the proposed methods for generating synthetic mouse trajectories. Section 4 describes our experimental framework (BeCAPTCHA-Mouse Benchmark) and presents the results obtained. Finally, Section 5 summarizes the conclusions and future works.

2. Mouse dynamics in the context of behavioral biometrics

Human-Machine interaction generates a heterogeneous flow of data from multiple channels. This interaction generates patterns affected by: humans (e.g. attitude, emotional state, neuromotor, and cognitive abilities), sensor characteristics (e.g. ergonomics, precision), and task characteristics (e.g. easy of use, design, usefulness). Modeling the user behavior using these heterogeneous data streams is an ongoing challenge with applications in a variety of fields such as security, e-health, gaming, or education [8,9,18]. Among this variety of data sources, in the present paper we concentrate in behavioral biometric signals [19].

Table 1
Biometric characteristics typically obtained in human-computer interaction. We rate each factor with * (low), ** (medium), and *** (high). Uniq = Uniqueness, Univ = Universality, Meas = Measurability, Perf = Performance, Circ = Circumvention, Acce = Acceptability, Cog = Cognitive, Neu = Neuromotor.

	Uniq.	Univ.	Meas.	Perf.	Circ.	Acce.	Cog.	Neu.
Keystroke	**	**	***	***	**	**	**	***
Stylometry	*	*	*	*	*	*	***	*
Web-log	**	*	***	**	*	*	***	*
Mouse	*	**	***	***	*	***	**	***

The literature of behavioral biometrics in the context of Human-Computer Interaction is large and includes several characteristics, e.g.: keystroking [20,21], handwriting [22], touchscreen signals [23,24], stylometry [25], and mouse dynamics [17]. Each characteristic has its pros and cons, therefore, a single biometric characteristic is usually not suitable for all applications. The biometric research community has identified several factors that determine the suitability of a biometric characteristic to be used in a certain application [7].

Table 1 rates these factors for biometrics characteristics typically obtained from Human-Computer Interaction highlighting Mouse Dynamics, the focus in the present paper. Note that we added two factors related to the nature of the patterns obtained from these characteristics (Cognitive and Neuromotor patterns) with respect to the characteristics defined by [7].

Now focusing in mouse dynamics for biometrics, in [17] researchers explored characteristics obtained from mouse tasks for user recognition. They analyzed 68 global features (e.g. duration, curvature, mean velocity) from mouse dynamics extracted during login sessions. Their results achieve up to 95% authentication accuracy for passwords with 15 digits. Besides, mouse dynamics can be combined with keystroke biometrics for continuous authentication schemes [26]. The fusion of both biometric modalities has been shown to outperform significantly each individual modality achieving up to 98% authentication accuracy [27]. In [28], the authors applied the Sigma-Lognormal Model based on the Kinematic Theory [15] to compress mouse trajectories. They suggested that mouse movements are the result of complex human motor control behaviors that can be decomposed in a sum of primal movements. In addition, in [29], the authors studied the relationship between eye gaze position and mouse cursor position on a computer screen

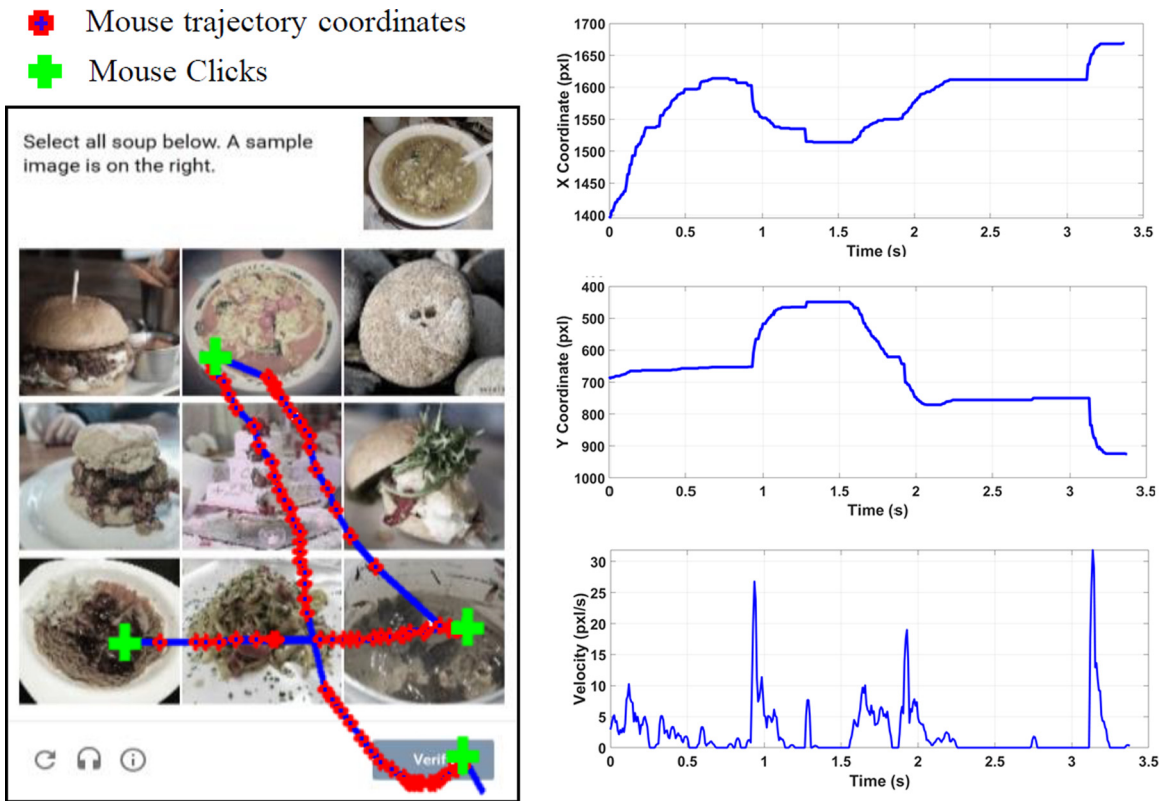


Fig. 2. An application example of our proposed mouse bot detection algorithm in combination with a traditional image-based CAPTCHA. While the user completes the image CAPTCHA task (cognitive challenge, left), our algorithm analyzes the mouse dynamics performed during the task ({**x**, **y**} coordinates and velocity profile, right).

during web browsing and suggested that there are regular patterns of eye/mouse movements associated to the motor cortex system.

3. BeCAPTCHA-Mouse: Bot detection based on mouse dynamics

The mouse is a very common device and its usage is ubiquitous in human-computer interfaces. Bot detection based on mouse dynamics can be therefore applied either in active or passive detectors.

Our BeCAPTCHA-Mouse bot detector is based on two main pillars: 1) we use mouse dynamics to extract neuromotor features capable to distinguish human behavior from bots (see Fig. 1); 2) we generate synthetic mouse trajectories to improve the learning framework of bot detectors[8].

Mouse dynamics are rich in patterns capable of describing neuromotor capacities of the users. Note that we do not claim to replace other approaches (e.g. Google's reCAPTCHA) by mouse-based bot detection, our purpose is to enhance them by exploiting the ancillary information provided by mouse dynamics (see Fig. 2).

Our proposed method for bot detection consists in characterizing each mouse trajectory (real and synthetic) with a fixed-size feature vector obtained from a neuromotor decomposition of the velocity profile, followed by a standard classifier. Each trajectory characterized in this way can be classified individually using standard classifiers into human or bot based on supervised training using a development groundtruth dataset. When multiple trajectories are available, standard information fusion techniques can be applied [30]. The more realistic the synthetic data used as groundtruth for training the classifier the stronger the classifier.

In our experimental work we demonstrate the effectiveness of the neuromotor features and the synthetic samples for different classifiers. The contribution and success of our BeCAPTCHA-Mouse bot detector is not in the particular classifier used, but in two other fronts (see Fig. 1): the high realism of the groundtruth data used for training our classifiers (with the methods presented in Section 3.2), and our proposed trajectory modeling using neuromotor features.

3.1. BeCAPTCHA-Mouse: Neuromotor analysis of mouse trajectories

By looking at typical mouse movements (see Fig. 3.a), we can observe some aspects typically performed by humans during mouse trajectories execution: an initial acceleration and final deceleration performed by the antagonist (activate the movement) and agonist muscles (opposing joint torque) [15], and a fine-correction in the direction at the end of the trajectory when the mouse cursor gets close to the click button (characterized by a low velocity that serves to improve the precision of the movement). These aspects motivated us to use neuromotor analysis to find distinctive features in human mouse movements. Neuromotor-fine skills, that are unique of human beings are difficult to emulate for bots and could provide distinctive features in order to tell humans and bots apart.

For this, we propose to model the trajectories according to the Sigma-Lognormal model [31] from the kinematic theory of rapid human movements [15]. The model states that the velocity profile of the human hand movements (mouse movements in this work) can be decomposed into primitive strokes with a Lognormal shape that describes well the nature of the hand movements ruled by the motor cortex. The velocity profile of these strokes is modeled as:

$$|\vec{v}_i(t)| = \frac{D_i}{\sqrt{2\pi}\sigma_i(t-t_{0i})} \exp\left(\frac{(\ln(t-t_{0i})-\mu_i)^2}{-2\sigma_i^2}\right) \quad (1)$$

[8] Note that mouse trajectories and mouse cursor are not technically the same. For simplification we will use the term mouse trajectory to refer to the information available to model the mouse movements (i.e., coordinates and timestamps).

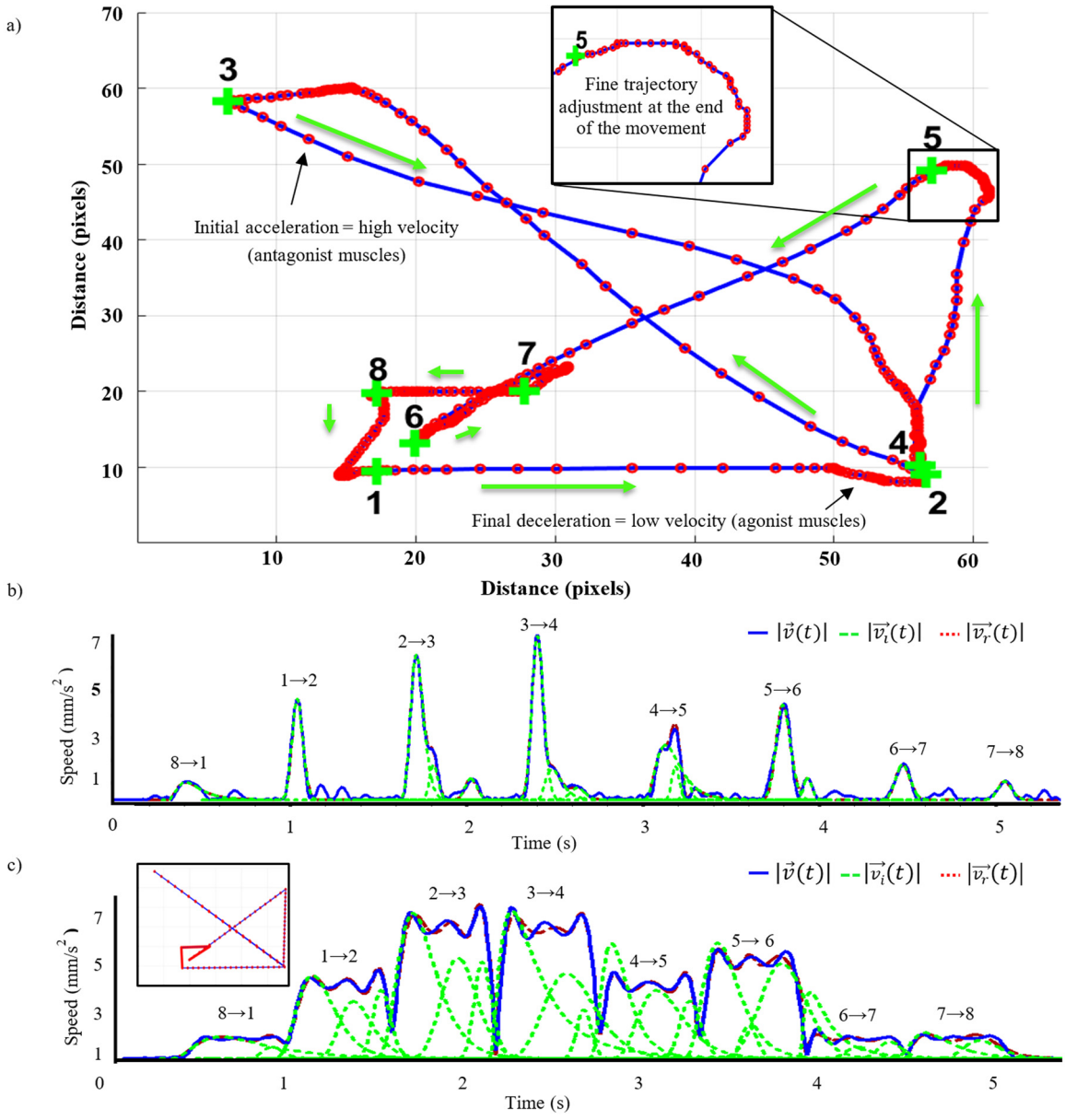


Fig. 3. a) Example of the mouse task determined by 8 keypoints: the crosses represent the keypoint where the user must click, red circles are the (x, y) coordinates obtained from the mouse device, and the black line is the mouse trajectory. b) and c) are examples of the Lognormal decomposition of a human mouse movement and a synthetic linear trajectory respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Sigma-Lognormal features description.

Parameter	Description
D_i	Input pulse: covered distance
t_{0i}	Initialization time: displacement in the time axis
μ_i	Log-temporal delay
σ_i	Impulse response time of the neuromotor system
θ_{si}	Starting angle of the stroke
θ_{ei}	Ending angle of the stroke

where the parameters are described in Table 2. The velocity profile of the entire hand movement is calculated as the sum of all these individual strokes:

$$\vec{v_r}(t) = \sum_{i=1}^{N} \vec{v_i}(t) \tag{2}$$

where N is the number of velocity strokes considered in the model. A complex action like handwriting signature or mouse movements, is a summation of these lognormals, each one characterized by the six parameters in Table 2. An example of this is shown in Fig. 3.b, where the black line is the velocity profile $|\vec{v}(t)|$ of the above human mouse task (Fig. 3.a), which is used as the input of the Sigma-Lognormal model. The green dashed lines correspond to the individual lognormal signals $|\vec{v_i}(t)|$ generated as in [31], which describes a method to automatically estimate both N and the parameters in Table 2 from an input trajectory $|\vec{v}(t)|$. Finally, the red dotted line $|\vec{v_r}(t)|$ is the reconstruction of the original velocity profile by summing all these generated individual lognormal signals. We can observe that the reconstructed signal matches almost perfectly with the original velocity profile of the human mouse movement, suggesting the potential of the Sigma-Lognormal model to describe neuromotor mouse movements. Lognormals with a high amplitude are typically observed

during the first part of the movement (agonist and antagonist activations), while smaller lognormals occur during the fine correction. The differences in lognormal sizes provide us information about the length of the trajectory (long trajectories have usually larger velocities).

The neuromotor feature set proposed for bot detection is computed from the six lognormal parameters described in Table 2. Each mouse trajectory generates N lognormal signals and each lognormal generates those 6 parameters from Table 2. For each parameter, we calculate 6 features: maximum, minimum, and mean for both halves of the trajectory. This is done because in natural mouse movements the lognormal parameters are usually very different between both halves of a given trajectory (e.g. Fig. 3.b). Additionally, we added the number of lognormals N that each mouse trajectory generates as an additional feature. This additional feature measures the complexity of the trajectory [32], having many lognormals means that the mouse trajectory has many changes in the velocity profile while few of them usually indicates more basic trajectories. As a result, the neuromotor feature set has size 37.

3.2. BeCAPTCHA-Mouse: Trajectory synthesis

In the present paper, a mouse movement is defined by the spatial trajectory across time between two consecutive clicks, i.e., a sequence of points $\{\mathbf{x}, \mathbf{y}\}$ and a velocity profile $|\vec{v}(t)|$, where $\mathbf{x} = [x_1, \ldots, x_M]$, $\mathbf{y} = [y_1, \ldots, y_M]$, and M is the number of time samples. A mouse trajectory is defined by two main characteristics: the shape (defined by $\{\mathbf{x}, \mathbf{y}\}$) and the velocity profile (defined by $|\vec{v}(t)|$). In order to generate realistic synthetic samples, both characteristics must be considered in the generation method. We propose two methods for synthetically generating such mouse movement.

3.2.1. Method 1: Function-based trajectories

We generate mouse trajectories according to three different trajectory shapes (linear, quadratic, and exponential) and three different velocity profiles (constant, logarithmic, and Gaussian). We can synthesize many different mouse trajectories that mimic human movements by varying the parameters of each function. To generate a synthetic trajectory $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$ with M points, first we define the initial point $[\hat{x}_1, \hat{y}_1]$ and ending point $[\hat{x}_M, \hat{y}_M]$. Second, we select one of three velocity profiles $\left|\vec{\hat{v}}(t)\right|$: *i)* constant velocity, where the distance between adjacent points is constant; *ii)* logarithmic velocity, where the distances are gradually increasing (acceleration); and *iii)* Gaussian velocity, in which the distances first increase and then decrease when they get close to the end of the trajectory (acceleration and deceleration). Third, we generate a sequence $\hat{\mathbf{x}}$ between $\hat{x}_1$ and $\hat{x}_M$ spaced according to the selected velocity profile. The $\hat{\mathbf{y}}$ sequence is then generated according to the shape function. For example, for a shape defined by the quadratic function $\hat{y} = a\hat{x}^2 + b\hat{x} + c$, we fit b and c for a fixed value of a by using the initial and ending points. We repeat the process fixing either b or c. The range of the parameters $\{a, b, c\}$ explored is determined by analyzing real mouse movements fitted to quadratic functions. Linear and exponential shapes are generated similarly.

Fig. 5 (trajectories D, E, and F) shows some examples of these mouse trajectories synthesized. That figure also shows the 3 different velocity profiles considered: the 3 trajectories in E have constant velocity, F shows acceleration (the distance between adjacent samples increases gradually), and D has initial acceleration and final deceleration. We can generate infinite mouse trajectories with this approach by varying the parameters of each function.

An important factor when synthetizing mouse trajectories is the number of points (M) of the trajectory. This usually varies depending not only on the length of the trajectory, but also on the direction, because different muscles are involved when we perform mouse trajectories in different directions. To emulate this phenomenon, we calculate the mean and standard deviation of the number of points for each of the 8 mouse trajectories from the human data used in the experiments. Then, we synthetize trajectories with different number of points following a Gaussian distribution with the calculated mean and standard deviation.

3.2.2. Method 2: GAN-based trajectories

For this approach we employ a GAN (Generative Adversarial Network) [33], in which two neuronal networks, commonly named Generator (defined by its parameters $\mathbf{w}_G$) and Discriminator (defined by its parameters $\mathbf{w}_D$), are trained one against the other (thus the 'adversarial'). The architecture of the GAN is depicted in Fig. 4. The aim of the Generator is to fool the Discriminator by generating fake trajectories $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$ very similar to the real ones $\{\mathbf{x}, \mathbf{y}\}$. We used a fixed sampling rate of 200Hz for all the real and generated trajectories. The sampling rate is determined by the real trajectories used in the learning framework (200Hz in our experiments). Therefore, the synthesized samples are generated with the same sampling rate. Other frequencies can be obtained subsampling the generated ones or re-training the GAN for a different sampling rate. The input of the Generator consist of a seed vector of R random numbers (in our experiments $R = 100$). The output of the Generator are two coordinate vectors $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$ with length equal to M (M can be fixed to generate different lengths). The input of the Discriminator consists of a batch including two types of trajectories: 1) *Bot*: synthetic trajectories generated by the Generator; 2) *Human*: real mouse trajectories chosen randomly from the Mouse DB described in next sections. The aim of the Discriminator is to predict whether the sample comes from the human set or is a fake created by the Generator. During the training phase, the GAN architecture will improve the ability of the Generator to fool the Discriminator. This architecture turns latent space points defined by the random seed into a classification decision: '*Bot*' (from the Generator) or '*Human*'. This learning process is guided by the real mouse trajectories from the Mouse DB. During the GAN training, the weights of the Discriminator ($\mathbf{w}_D$) remain frozen. The iterative training process will update the weights $\mathbf{w}_G$ of the Generator in a way that makes Discriminator more likely to predict '*Human*' when looking at synthetic mouse trajectories. If the Discriminator is not frozen during this process, it will tend to predict '*Human*' for all samples. The Discriminator is trained (weights $\mathbf{w}_D$ updated) after the update of the weights of the Generator ($\mathbf{w}_G$). This process is repeated iteratively (50 epochs in our experiments). Once the Generator is trained this way, then we can use it to synthesize mouse trajectories very similar to the human ones.

The topology employed in the Discriminator consist of two LSTM (Long Short-Term Memory) layers (with 128 and 64 units respectively, with '*LeakyReLU*' activation) followed by a dense layer (with 1 unit and '*Sigmoid*' activation). The dense layer of the Discriminator is used as a classification layer to distinguish between bot and real mouse trajectories ('*Binary Cross-Entropy*' loss function). For the Generator, we employ two LSTM layers (with 128 and 64 units respectively, with '*ReLU*' activation) followed by a dense layer with 2 units (one unit for build each $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$ mouse coordinates) and '*TimeDistributed*' activation.

The GAN architecture used in BeCAPTCHA follows the traditional approach proposed in [34], with an specific topology developed to capture the characteristics of mouse trajectories. For comparison with other state-of-the-art Generative Adversarial Architectures, we also added an implementation of the Time-GAN (TGAN), firstly proposed in [35]. According to the authors, TGAN models are better situated to work with time series data. The addition of the Embedding network jointly with a supervised loss function helps in the mapping process between the data features and latent space points during the adversarial training. Thanks to this, the TGAN ar-

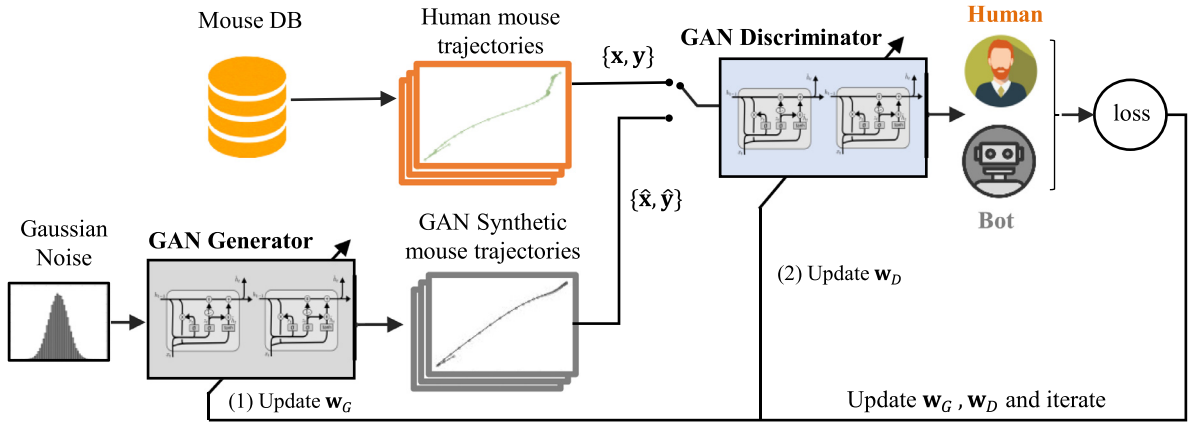


Fig. 4. The proposed architecture to train a GAN Generator of synthetic mouse trajectories. The Generator learns the human features of the mouse trajectories and generate human-like ones from Gaussian Noise. Note that the weights of the Discriminator $\mathbf{w}_D$ are trained after the update of the weights of the Generator $\mathbf{w}_G$.

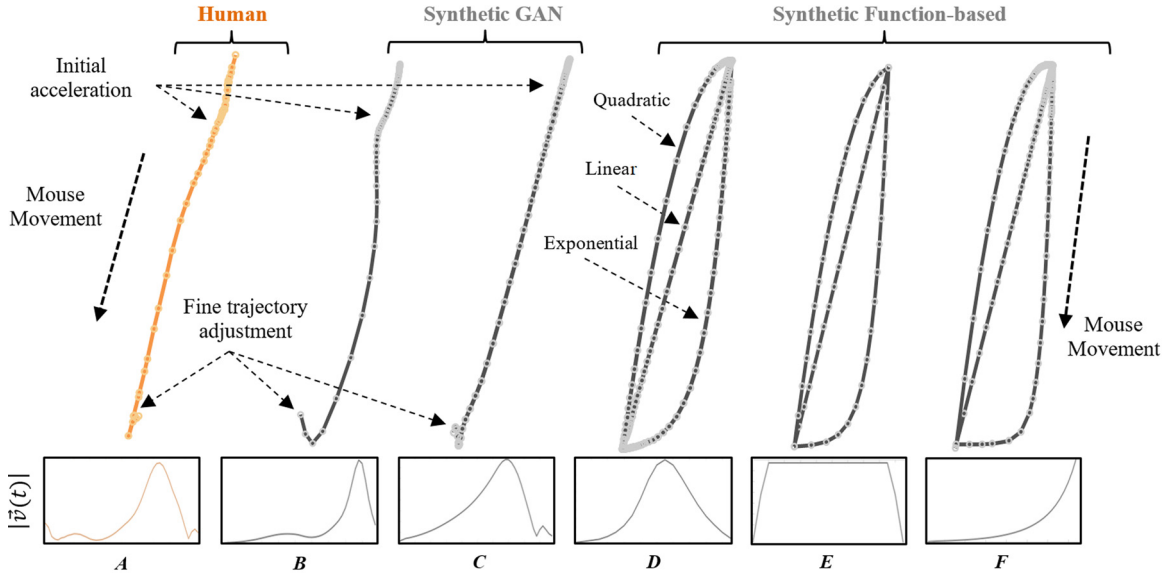


Fig. 5. Examples of mouse trajectories and their velocity profiles employed in this work: *A* is a real one extracted from a task of the database; *B* and *C* are synthetic trajectories generated with the GAN network; *D*, *E* and *F* are generated with the Function-based approach. Note that for each velocity profile (*D* = Gaussian, *E* = constant, *F* = logarithmic), we include the three Function-based trajectories (linear, quadratic, and exponential).

chitecture is able to learn the underlying temporal dynamics of the data to generate more realistic synthetic samples. For our implementation of TGAN we keep the same typologies for both Generator and Discriminator as done in our traditional GAN to allow for a fair comparison.

Both GAN and TGAN architectures were trained using 60% of the human mouse trajectories in the database. Training details: learning rate $\alpha = 2 \times 10^{-4}$, Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, 50 epochs with a batch size of 128 samples for both Generator and Discriminator. The architectures were implemented in `Python` with `Keras-Tensorflow` libraries.

Figure 5 shows two examples (trajectories *B* and *C*) of synthetic mouse trajectories generated with the GAN network and the comparison with a real one. We can observe high similarity between the two synthetic examples and the real one. Human mouse patterns such us the initial acceleration and the final trajectory fine correction that we discussed before are automatically learned by the GAN network and reproduced in the synthetic trajectories generated.

Figure 6 shows six feature distributions obtained from human and synthetic trajectories. The distributions comprise three features from the Sigma-Lognormal set and other three from the global feature set [16]. The feature set proposed in [16] consists of 6 global features: duration, distance, displacement, average angle, average velocity, and move efficiency (distance over displacement). The distributions obtained from the synthetic samples showed characteristics similar to the human ones. The larger differences can be seen in the feature distance, where the human samples showed two distributions for short and long trajectories. Note that these features can be modified during the generation method to produce trajectories with distances similar to the target distributions.

4. Experiments

4.1. BeCAPTCHA-Mouse benchmark: Database

The human mouse trajectories employed in this work were extracted from Shen et al. database [36], which is comprised of more than 200K mouse trajectories acquired from 58 users who completed 300 repetitions of the task. Acquisition of data from each subject took between 30 days and 90 days. In each repetition, the task was to click 8 buttons that appeared in the screen sequentially. This task was repeated twice in each session. Fig. 3.a shows an example of the whole mouse movement task. Note that the buttons are placed in a particular order to generate mouse trajec-

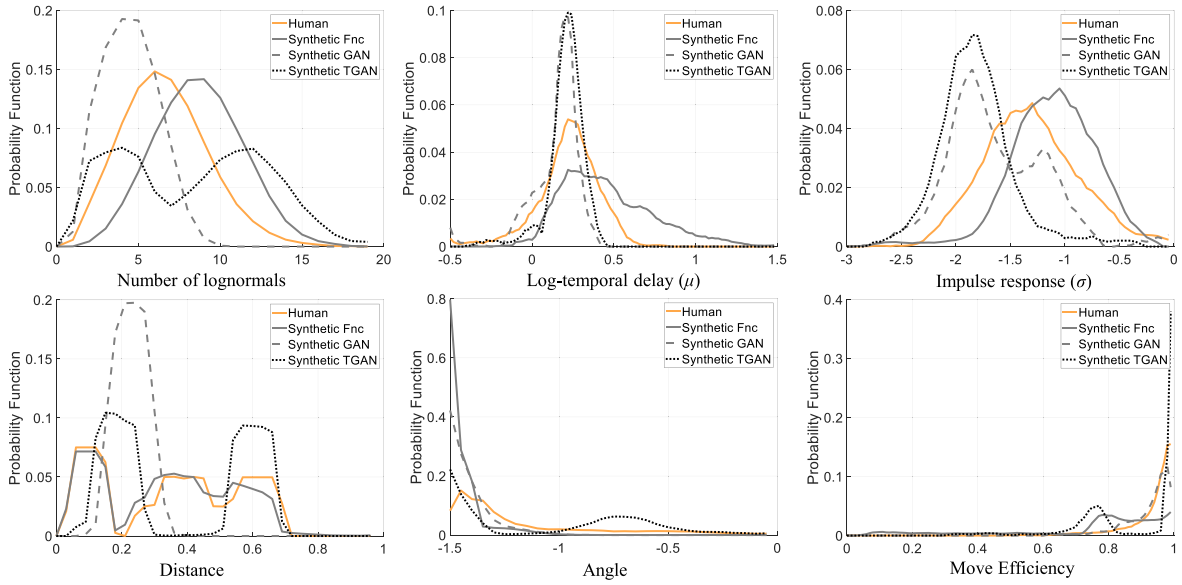


Fig. 6. Probability distributions of six features from: (Up) Lognormal feature set and (Down) global feature set [16]) for Human, function-based, GAN, and TGAN mouse trajectories.

Table 3
Bot detection accuracy (%) obtained from all the 8 trajectories for each of the feature sets. VP (Velocity Profile): VP1 = constant velocity, VP2 = initial acceleration, VP3 = initial acceleration and final deceleration.

	Bot: Function-based										
	Linear			Quadratic			Logarithmic				
Feature Set	VP1	VP2	VP3	VP1	VP2	VP3	VP1	VP2	VP3	Bot:GAN	Bot:TGAN
Global [16]	99.7	99.6	99.7	95.3	96.7	96.8	97.2	96.5	97.3	**99.8**	**99.7**
Neuromotor	99.1	98.7	99.3	96.9	96.3	94.7	96.3	95.2	94.7	98.0	95.6
Ours	**99.9**	**99.7**	**99.8**	**98.0**	**99.0**	**98.4**	**98.2**	**98.9**	**98.9**	99.7	99.6

tories with different directions (rightwards, upwards, downwards, and oblique) and different lengths.

In the present work, we define a mouse trajectory as the mouse displacement that occurs between two click buttons. Therefore, the mouse movement task of Fig. 3.a is composed of 8 mouse trajectories. The raw data recorded during the acquisition process was: the mouse position over the screen ({**x**, **y**} axis position in pixels), the event (movement or click), and timestamp of the event. The experiments presented in this work are performed using a subset of the database including 35 samples (randomly chosen) from each of the 58 users available (more than 5K trajectories in total).

Figure 3 (c) shows the decomposition of a synthetic function-based trajectory with linear shape. We can observe the huge differences between both lognormal decompositions (the human trajectory and the synthetic one) by looking at the shape of the lognormal signals. The synthetic trajectory has wider lognormals and they are more symmetric than the human ones. Note that the Sigma-Lognormal algorithm introduces a low-pass filter to the input signal, that is the reason why the velocity profile of the synthetic trajectory (Fig. 3.c) is a bit smoothed, but the difference between both synthetic and human velocity profiles is still patent.

The BeCAPTCHA-Mouse Benchmark is composed of 5K human trajectories and 10K synthetic trajectories generated according to the two methods proposed (5K function-Based and 5K GAN trajectories). Both real and synthesized samples are characterized by a variety of lengths, directions, and velocities.

4.2. Results and comparison with previous approaches

The main contributions of this work are: 1) a novel feature set based on the combination of global and neuromotor characteristics of the mouse trajectories; 2) two methods to generate synthetic mouse trajectories for improving training and evaluation of bot detection methods.

To validate the first contribution, we have extracted the proposed feature set from human and synthetic mouse trajectories. For this first experiment, we use a Random Forest (RF) classifier because of its best performance among all classifiers evaluated (as we will see in the next section). For each RF, we train the classifier by using 70% of all samples (up to 1,500 samples available for each type of trajectory between both synthetic and real ones) randomly chosen as the training set. The other 30% samples are employed for evaluation. The results are obtained by repeating each experiment 5 times and averaging, with a standard deviation of $\sigma \sim 0.1\%$. All classifiers employed in this section were implemented in `Python` with the `scikit-learn` library.

The first experiment is aimed to demonstrate the performance of the proposed feature set. The Table 3 present the results when features from all 8 trajectories are combined (each RF is trained using features from all 8 trajectories). Additionally, we compare the performance achieved with existing approaches [16]. The feature set proposed in [16] consists of 6 global features: duration, distance, displacement, average angle, average velocity, and move efficiency (distance over displacement). The results suggest that the feature set proposed in [16] outperforms the neuromotor features proposed here only for GAN and TGAN synthetic trajectories. The best performance is obtained overall with an extended set composed by both sets of features. The extended set has the best results with an average around 99% of accuracy independently of the type of synthetic trajectory.

The second experiment is aimed to demonstrate the performance of classifiers when training with both human and bot sam-

Table 4
Bot detection accuracy (%) of the different feature sets for models trained with and without synthetic samples (bots) and evaluated using human samples and bots samples. SVM (SVM with Radial Basis Function kernel), RF (Random Forest), KNN (K-Nearest Neighbors), F (bots generated with the function-based method), G (bots generated with the GAN method).

	Training Samples											
	Only Humans [16]			Humans+Bots (F)			Humans+Bots (G)			Humans+Bots (F,G)		
Feature Set	SVM	RF	KNN	SVM	RF	KNN	SVM	RF	KNN	SVM	RF	KNN
Global [16]	63.5	57.1	53.4	65.7	57.3	62.1	51.6	59.1	52.0	96.6	99.5	98.2
Neuromotor	60.0	60.3	52.4	54.1	62.8	63.5	58.4	58.3	60.2	97.2	97.3	93.0
Ours	**65.2**	62.0	53.7	**65.8**	61.0	64.1	60.1	**63.0**	61.2	98.2	**99.7**	96.8

Table 5
Bot detection accuracy (%) obtained for each of the 8 trajectories and the Neuromotor feature set. The accuracies were obtained using the same RF classifiers employed for Table 3. VP (Velocity Profile): VP1 = constant velocity, VP2 = initial acceleration, VP3 = initial acceleration and final deceleration.

	Bot: Function-based										
	Linear			Quadratic			Logarithmic				
Trajectories	VP1	VP2	VP3	VP1	VP2	VP3	VP1	VP2	VP3	Bot: GAN	Bot: TGAN
8 → 1	98.6	96.3	99.0	91.0	91.0	92.3	89.0	88.6	89.3	96.9	97.4
1 → 2	99.7	98.6	97.2	91.6	98.3	92.2	95.8	92.3	92.5	96.7	97.7
2 → 3	99.4	99.1	99.7	95.3	96.4	88.0	94.4	98.9	90.5	99.9	93.9
3 → 4	99.7	97.5	97.0	94.2	96.6	90.5	94.2	95.1	93.0	99.7	94.0
4 → 5	99.9	98.0	99.4	95.5	94.7	92.5	93.9	95.4	93.9	97.0	95.4
5 → 6	99.9	98.9	99.1	92.8	97.5	91.4	93.3	95.1	94.4	98.3	95.5
6 → 7	99.1	98.3	98.6	90.2	89.7	93.6	88.8	92.3	93.6	98.1	98.0
7 → 8	97.0	96.6	97.5	92.2	93.3	93.0	88.3	88.6	93.1	98.7	98.1

Table 6
Bot detection performance metrics in % (Acc = Accuracy, AUC = Area Under the Curve, Pre = Precision, Re = Recall, and F1) for the different scenarios: Function-based, GAN, and Combination.

	Bot Generation Method														
	Function-based					GAN					Function-based + GAN				
Classifiers	Acc	AUC	Pre	Re	F1	Acc	AUC	Pre	Re	F1	Acc	AUC	Pre	Re	F1
SVM	98.0	99.4	98.6	96.7	97.7	98.5	99.6	99.2	97.9	98.5	98.2	99.4	97.3	99.0	97.4
KNN	93.4	98.1	93.6	93.2	93.5	94.1	99.4	99.8	88.3	93.6	92.0	97.4	90.7	93.2	92.1
RF	**98.5**	**99.8**	**98.6**	**98.8**	**98.7**	**99.7**	**99.9**	**99.5**	**99.9**	**99.7**	**98.7**	**99.9**	**98.8**	**99.0**	**99.0**
MLP	94.6	94.1	95.0	94.2	94.6	93.4	93.5	95.4	92.3	93.9	92.2	91.5	89.8	95.4	92.5
LSTM	98.2	99.8	97.6	98.8	98.2	99.2	98.0	99.7	98.9	99.5	97.3	99.7	96.7	97.9	97.3
GRU	98.4	99.4	98.5	98.6	98.6	99.3	99.2	99.2	90.2	99.0	99.8	99.8	94.4	99.0	96.9

ples generated with the proposed methods (second contribution). For this experiment we propose three different bot detection scenarios according to the data employed to train and evaluate the bot detection approaches: *i)* training only with the real samples (usually referred as anomaly detector), *ii)* employing one type of synthetic samples for training and the other one for testing (agnostic classification), and *iii)* employing the real and both kinds of synthetic samples to train and test. The aim of the experiment is to evaluate to what extent the inclusion of synthetic samples in the learning framework serves to improve the accuracy of the model in comparison with previous methods based only on human data [16]. For this experiment we included the three classifiers with the best performances reported in [16]: SVM with Radial Basis Function (RBF) kernel, Random Forest (RF) and K-Nearest Neighbors (KNN). Note that the bot detection method proposed in [16] was based exclusively on human samples trained as an anomaly detector. In this work, we explore new learning frameworks using both human and bot samples during training and evaluation.

Table 4 shows the bot detection accuracy for the different scenarios depending of the training data employed. As in the previous experiment, the classifiers are trained using trajectories from all 8 directions and synthetic samples from all 10 types of attacks. The results show that the synthetic samples and the feature set proposed in this work allows to reduce the error by 95.4% in comparison with the previous existing method based exclusively on human samples [16]. As can be seen, the classifier trained only with real samples was not capable to detect most of the attacks with accuracy rates lower than 70% either for global features and neuromotor features. In the agnostic classification, the poor results achieved when training with one type of synthetic samples and testing with the other one suggest there is a huge complementarity among both generation methods. These results suggest that future addition of other synthetic generation methods could improve the performance of bot detectors. The importance of synthetic samples is twofold: *i)* evaluation of bot detection algorithms under challenging attacks generated according to different methods; and *ii)* training better detectors to model both human and synthetic behaviors. The results in Table 4 show the potential of the synthetic samples and its usefulness to train better models capable to deal with all types of attacks.

4.3. BeCAPTCHA-Mouse: Ablation study

4.3.1. Influence of trajectory characteristics

In the first experiment we analyze the impact of the different human mouse trajectories in the classification performance. The

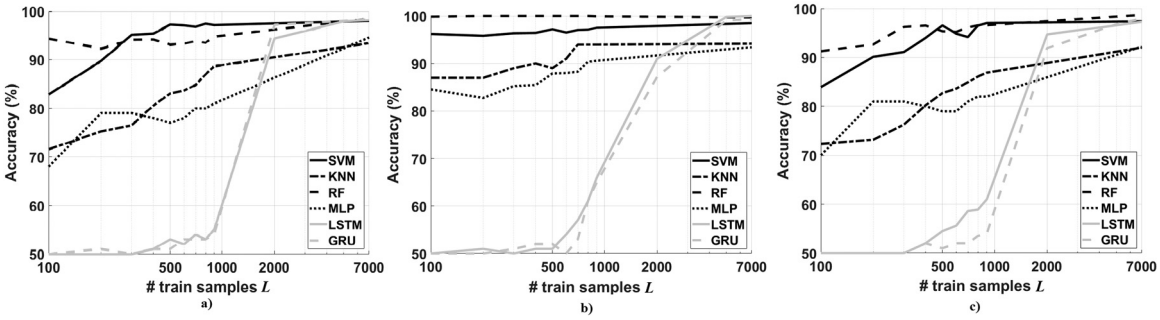


Fig. 7. Accuracy curves (%) against the number of train samples ($100 \leq L \leq 7,000$) to train the different classifiers in Function-based (a), GAN (b), and Combination (c) classification scenarios.

Table 7
Performance metrics in % (AUC = Area Under the Curve, Acc, Pre, Re, and F1) for the different setups of GAN Discriminator in bot detection. In brackets the number of neurons for the first/second LSTM layer respectively used in the Discriminator.

	Bot Generation Method														
	Function-based					GAN					Function-based + GAN				
Discriminators	Acc	AUC	Pre	Re	F1	Acc	AUC	Pre	Re	F1	Acc	AUC	Pre	Re	F1
LSTM (128/64)	**89.9**	**93.2**	**88.5**	**90.0**	**89.3**	96.8	99.6	95.0	98.7	96.8	**89.6**	**93.9**	**89.2**	**90.0**	**89.6**
LSTM (64/32)	74.0	72.1	67.0	95.6	78.7	**99.9**	**99.9**	**99.9**	**99.9**	**99.9**	73.0	76.1	65.9	96.0	78.1
LSTM (32/16)	81.4	80.2	77.9	88.0	82.6	99.7	98.9	99.6	99.9	99.8	78.8	76.0	74.4	88.0	80.6
LSTM (16/8)	56.8	58.6	54.2	86.8	66.7	56.2	91.3	53.3	99.9	69.5	64.0	67.0	59.5	87.2	70.7

experiments are divided according to the 8 real mouse trajectories present in the whole task. This means that we classify at trajectory level (i.e. the mouse trajectory performed between two consecutive click buttons) instead of classifying the whole task. This is because the task was designed to take into account trajectories with different directions and lengths, and therefore, different muscles configurations are involved in each trajectory. In this way, we can analyze which mouse trajectories are better to discriminate between humans and bots. We train 11×8 different RFs (one for each type of attack and mouse trajectory, see columns in Table 5) using both human and synthetic trajectories.‘

Table 5 shows the results for the different bot generation methods and the 8 trajectories derived from the movements between the 8 keypoints (plotted in Fig. 3.a). The table shows the bot classification accuracy in % (human vs bot). First, comparing among the different trajectories, we can observe that the shorter ones ($8 \rightarrow 1$, $6 \rightarrow 7$, and $7 \rightarrow 8$) show higher classification errors compared to the larger ones. Short trajectories generate less neuromotor information: initial acceleration, final deceleration, and trajectory corrections are less pronounced in short trajectories. Second, logarithmic trajectory shapes achieve the worst classification performance, as we expected, because the shape of logarithmic functions fit better the human trajectories shapes. Third, the most significant parameter when synthetizing trajectories is the velocity profile. When VP = 3 (i.e., initial acceleration and final deceleration), the synthetic trajectories are able to fool the classifier up to 17% of the times. This confirms that the velocity profile of human mouse trajectories plays and important role when describing human features in mouse dynamics. Four, the GAN and TGAN Generators (last two columns in Table 5) result in lower classification errors compared with the function-based method. This is surprising after visualizing the high similarity between human and GAN-generated trajectories (see Fig. 5 *A* vs *B* and *A* vs *C*). We interpret this result with care: on the one hand it demonstrates that our bot detection approach is powerful against realistic and sophisticate fakes, but on the other hand both GAN and TGAN Generators can be improved to better fool our detector. Although the synthetic samples generated with them seems very realistic to the human eye, the RF classifiers were capable of detecting synthetic samples with high accuracy. These high classification rates suggest that adversarial learning Generators introduce patterns that allow its detection [33].

4.3.2. Influence of the classifier

For the following experiments, we performed an ablation study on different classifiers to analyze their performance in bot detection for the different bot generation methods proposed in this work: Function-Based, GAN, and their combination. It is worth noting that all classifiers are trained using trajectories from all 8 directions and synthetic samples from all 10 types of attacks, as reported in Table 4 to allow fair comparisons.

Table 6 shows the performance of classification algorithms: Support Vector Machine (SVM) with a Radial Basis Function (RBF), K-Nearest Neighbors (KNN) with $k = 10$, Random Forest (RF), Multi-Layer Perceptron (MLP), and 2 Recurrent Neuronal Networks (RNN), (one composed by Long Short-Term Memory (LSTM) units and the other with Gated Recurrent Units (GRU). The RNNs (i.e. LSTM and GRU) were trained directly with the raw data (i.e. the sequence of points $\{\mathbf{x}, \mathbf{y}\}$ of the mouse trajectories) instead of extracting the global features (i.e. Neuromotor + Baseline [16]) as done with the statistical classifiers. The RNNs have the same architecture as the Discriminator of the GAN: two recurrent layers of 128 and 64 units respectively, followed by a dense layer to classify between fake and real mouse trajectories. All classifiers were trained and tested following the same experimental protocol as in Section 4.2, using 70% of all samples (up to 10K samples between both real and synthetic samples when combining all types of trajectories) randomly chosen as the training set (named L in this section, with $L = 7,000$). The results are reported in terms of Accuracy, AUC (Area Under the Curve), Precision, Recall, and F1.

First, we can observe that the best results among the statistical classifiers are achieved by the RF classifier followed by the SVM. KNN and MLP perform worst, although all classifiers have accuracy rates over 90%. Secondly, among the different RNNs, the configuration with LSTM units performs sightly better than the one with GRU units, even though both recurrent network setups are outperformed by the RF classifier. These results suggest that the feature set chosen to train and test the statistical classifiers is suitable for the mouse bot detection task, outperforming other approaches

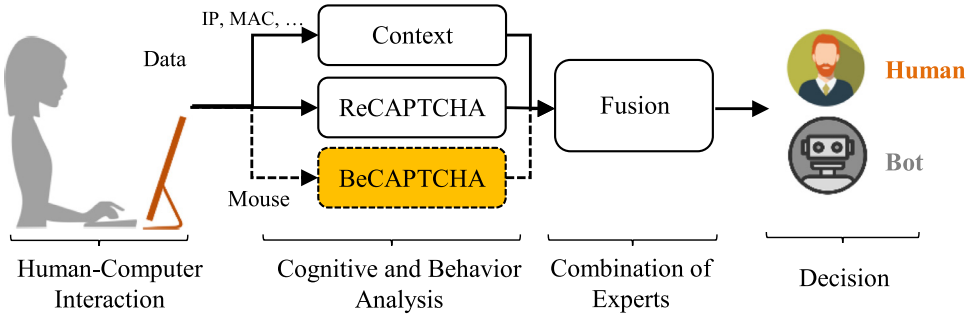


Fig. 8. Block diagram of multimodal bot detection. The response of the bot detector is a combination of responses from different experts. The bot detector proposed in this work can be used independently or in combination with existing bot detectors.

based on deep neuronal networks architectures. Nonetheless, the RNNs demonstrate its capacity to extract useful features from the raw data.

4.3.3. Influence of the number of training samples

In the next experiment we explore whether the number of training samples (L) plays and important role in the classification performance. We want to highlight that the training and the evaluation sets have the same number of human (L_h) and synthetic (L_s) samples, i.e.: $L_h = L_s = L/2$.

For this, in Fig. 7 we plot the accuracy curves of the previous classifiers according to the number of samples employed in their training set. As expected, the accuracy improves in all scenarios when we enlarge the number of train samples. However, there are important differences between the statistical and the RNNs approaches. Meanwhile all statistical classifiers achieve their maximum performance with $L = 500$, both LSTM and GRU are not able to reach the same performance with only 500 train samples. In fact, they need at least $L = 2,000$ to perform as well as the statistical classifiers. This shows the superior performance of the statistical classifiers in those scenarios where the number of samples to train the classifiers are scarce.

4.3.4. Using the GAN discriminator as classifier

Finally, in the last experiment we replaced the previously introduced RNNs classifiers by the Discriminator model of the GAN architecture. The idea is to analyze in what extent the Discriminator of the GAN Network trained only with the synthetic samples generated by the Generator (and the real ones) during the GAN training could perform better in classification than the previous RNNs trained from scratch. For this, we tuned the number of neurons of the two LSTM layers of the Discriminator and trained a new GAN network for each Discriminator setup proposed.

Table 7 shows the performance of 4 GAN Discriminator setups for the 3 classification scenarios proposed: the function-based, GAN, and their Combination. As we expected, the performance using GAN classification is much better than the performance achieved by the LSTM and GRU networks of the previous experiment, due to the Discriminators were trained specifically to discriminate between the synthetic mouse trajectories generated by the GAN Generator and the human ones. However, the Discriminators also classify quite well in the function-based scenario, even though no Function-based sample was employed to train them ($L_s = 0$). In fact, as we increase the complexity of the Discriminator with more neurons in both layers, the performance improves up to 90% of accuracy, close to the results achieved by the LSTM and GRU networks trained with $L_s = 7,000$ samples. These results show the potential of the GAN architecture, not only to generate synthetic mouse trajectories with similar shape to the human ones with the Generator, but also for classification purposes, as the Discriminator is able to classify between human and bot trajectories even against synthetic trajectories not seen during the training phase.

5. Conclusions and future work

We have explored behavioral biometrics for bot detection during human-computer interaction. In particular, we have analyzed the capacity of mouse dynamics to describe human neuromotor features. Our conclusions in comparison to state-of-the-art works suggest that there is unexploited potential of mouse dynamics as a behavioral biometric for tasks such as bot detection.

In particular, we have proposed BeCAPTCHA-Mouse, a bot detection algorithm based on mouse dynamics, and a related benchmark[9], the first one public for research in bot detection and other mouse-based research areas including HCI, security, and human behavior.

The proposed approach is able to discriminate between humans and bots with up to 98.7% of accuracy, even with bots of high realism, and only one mouse trajectory as input. This proves the potential of mouse dynamics for Turing tests. Additionally, we also provided an exhaustive ablation study on different classifiers to explore the capacity of these algorithms for the bot detection task. Random Forests (RF) have demonstrated to perform the best in all scenarios evaluated followed by an LSTM network. However, when the number of train samples is reduced ($L \leq 1,000$), the LSTM is not able to classify as well as the RF classifier. In fact, the LSTM can be replaced by the Discriminator of the GAN network when the lack of bot samples to train the system makes the deep learning approaches unavailable, showing a superior performance even against bot samples not seen during the training phase. This results suggest that the GAN architecture is a powerful tool not only to generate human-like mouse trajectories, but also to detect bot samples from other synthetic generation methods.

As future work, we aim at improving the neuromotor feature set by calculating secondary features inferred from the main ones. Also, we propose to improve the GAN model in two ways: *i)* combine both synthesis methods by using the function-based trajectories as the input of the GAN model instead of Gaussian noise, and *ii)* experimenting with different amount of layers/units in the GAN Generator to increase the variety of the synthetic mouse trajectories generated. Both techniques could generate more sophisticate and human-like trajectories. Finally, in this paper we only considered mouse trajectories acquired from mouse devices. We also propose to analyze mouse-pad trajectories normally performed when using laptops as another line of research.

The exploitation of behavioral biometrics for bot detection is an open research line with large opportunities and challenges. These challenges include the study of other ways of interaction beyond

[9] https://github.com/BiDAlab/BeCAPTCHA-Mouse

mouse such as keystroking [20,21] or touchscreen gestures [23] for bot detection, and their application to mobile scenarios [14]. We want to highlight that behavioral CAPTCHAs are compatible with previous CAPTCHA technologies and it could be added as a new cue to improve existing bot detection schemes in a multiple classifier combination [30] (see Fig. 8).

Recent fusion techniques incorporating contextual information [30] will be also explored for improving BeCAPTCHA. Finally, we'll try to improve our methods taking advantage of existing large-scale human-computer interaction datasets [24] and existing models [37] by using transfer learning methods [38].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

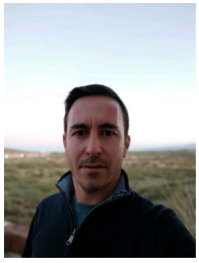
Acknowledgements

This work has been supported by projects: TRESPASS-ETN (MSCA-ITN-2019-860813), PRIMA (MSCA-ITN-2019-860315), BIBECA (RTI2018-101248-B-I00 MINECO), and by BBforTAI (PID2021-127641OB-I00 MICINN/FEDER). A. Acien is supported by a FPI fellowship from the Spanish MINECO.

References

[1] A.P. Saygin, I. Cicekli, V. Akman, Turing test: 50 years later, Minds and Machines 10 (2000) 463–518.
[2] Y. Gorodnichenko, T. Pham, O. Talavera, Social Media, Sentiment and Public Opinions: Evidence from Brexit and USElection, Working Paper, National Bureau of Economic Research, 2018.
[3] Y. Xie, S.-Z. Yu, A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors, IEEE/ACM Trans. Netw. 17 (2009) 54–65.
[4] E. Bursztein, M. Martin, J. Mitchell, Text-based CAPTCHA strengths and weaknesses, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, in: CCS –11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 125–138.
[5] K. Bock, D. Patel, G. Hughey, D. Levin, unCaptcha: A low-resource defeat of reCaptcha's audio challenge, 11th USENIX Workshop on Offensive Technologies (WOOT 17), USENIX Association, Vancouver, BC, 2017.
[6] I. Akrout, A. Feriani, Akrout, Hacking google reCAPTCHA v3 using reinforcement learning, in: Conference on Reinforcement Learning and Decision Making, 2019.
[7] A.K. Jain, K. Nandakumar, A. Ross, 50 Years of biometric research: accomplishments, challenges, and opportunities, Pattern Recognit Lett 79 (2016) 80–105.
[8] E. Nosakhare, R. Picard, Toward assessing and recommending combinations of behaviors for improving health and well-being, ACM Trans. Comput. Healthcare 1 (1) (2020).
[9] J. Hernandez-Ortega, R. Daza, A. Morales, J. Fierrez, J. Ortega-Garcia, edbb: Biometrics and behavior for assessing remote education, AAAI Workshop on Artificial Intelligence for Education (AI4EDU), 2020.
[10] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, J. Ortega-Garcia, Presentation attacks in signature biometrics: types and introduction to attack detection, in: S. Marcel, M. Nixon, J. Fierrez, N. Evans (Eds.), Handbook of Biometric Anti-Spoofing, Springer, 2019, pp. 439–453.
[11] A. Acien, A. Morales, R. Vera-Rodriguez, J. Fierrez, Multilock: Mobile active authentication based on multiple biometric and behavioral patterns, in: Proc. ACM Intl. Conf. on Multimedia, Workshop on Multimodal Understanding and Learning for Embodied Applications (MULEA), 2019, pp. 53–59.
[12] L. Ogiela, M.R. Ogiela, Cognitive security paradigm for cloud computing applications, Concurrency and Computation: Practice and Experience 32 (8) (2020) e5316.
[13] M.R. Ogiela, L. Ogiela, Cognitive keys in personalized cryptography, in: 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), IEEE Computer Society, 2017, pp. 1050–1054.
[14] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, O. Delgado-Mohatar, BeCAPTCHA: behavioral bot detection using touchscreen and mobile sensors benchmarked on humidb, Eng Appl Artif Intell 98 (2021) 104058.
[15] R. Plamondon, A kinematic theory of rapid human movements, Biol Cybern 72 (4) (1995) 295–307.
[16] Z. Chu, S. Gianvecchio, H. Wang, Bot or human? a behavior-based online bot detection System, in: P. Samarati, I. Ray, I. Ray (Eds.), From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday, Springer International Publishing, Cham, 2018, pp. 432–449.
[17] A.A.E. Ahmed, I. Traore, A new biometric technology based on mouse dynamics, IEEE Trans Dependable Secure Comput 4 (3) (2007) 165–179.
[18] H. Shrobe, D.L. Shrier, A. Pentland, Behavioral biometrics, in: New Solutions for Cybersecurity, 2018, pp. 365–377.
[19] A. Alzubaidi, J. Kalita, Authentication of smartphone users using behavioral biometrics, IEEE Communications Surveys Tutorials 18 (3) (2016) 1998–2026.
[20] A. Morales, J. Fierrez, R. Tolosana, J. Ortega-Garcia, J. Galbally, M. Gomez-Barrero, A. Anjos, S. Marcel, Keystroke biometrics ongoing competition, IEEE Access 4 (2016) 7736–7746.
[21] A. Acien, J.V. Monaco, A. Morales, R. Vera-Rodriguez, J. Fierrez, TypeNet: Scaling up keystroke biometrics, in: Proc. IEEE/IAPR International Joint Conference on Biometrics (IJCB), 2020.
[22] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, J. Ortega-Garcia, DeepSign: Deep on–line signature verification, IEEE Transactions on Biometrics, Behavior, and Identity Science 3 (2021) 1–11.
[23] J. Fierrez, A. Pozo, M. Martinez-Diaz, J. Galbally, A. Morales, Benchmarking touchscreen biometrics for mobile authentication, IEEE Trans. on Information Forensics and Security 13 (11) (2018) 2720–2733.
[24] A. Acien, A. Morales, R. Vera-Rodriguez, J. Fierrez, Smartphone sensors for modeling human-computer interaction: General outlook and research datasets for user authentication, IEEE Intl. Workshop on Consumer Devices and Systems (CDS), 2020.
[25] H. Locklear, S. Govindarajan, Z. Sitov, A. Goodkind, D.G. Brizan, A. Rosenberg, V.V. Phoha, P. Gasti, K.S. Balagani, Continuous authentication with cognition–centric text production and revision features, in: Proceedings of the IEEE International Joint Conference on Biometrics, 2014, pp. 1–8.
[26] T. Sim, S. Zhang, R. Janakiraman, S. Kumar, Continuous verification using multimodal biometrics, IEEE Trans Pattern Anal Mach Intell 29 (4) (2007) 687–700.
[27] S. Mondal, P. Bours, A study on continuous authentication using a combination of keystroke and mouse biometrics, Neurocomputing 230 (2017) 1–22.
[28] D. Martín-Albo, L.A. Leiva, J. Huang, R. Plamondon, Strokes of insight, Inf. Process. Manage. 52 (6) (2016) 989–1003.
[29] M.C. Chen, J.R. Anderson, M.H. Sohn, What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing, in: Proceedings of the CHI –01 Extended Abstracts on Human Factors in Computing Systems, in: CHI EA –01, 2001, pp. 281–282. New York, NY, USA
[30] J. Fierrez, A. Morales, R. Vera-Rodriguez, D. Camacho, Multiple classifiers in biometrics. Part 1: fundamentals and review, Information Fusion 44 (2018) 57–64.
[31] A. Fischer, R. Plamondon, Signature verification based on the kinematic theory of rapid human movements, IEEE Trans Hum Mach Syst 47 (2) (2017) 169–180.
[32] R. Vera-Rodriguez, R. Tolosana, J. Hernandez-Ortega, A. Acien, A. Morales, J. Fierrez, J. Ortega-Garcia, Modeling the complexity of signature and touch-screen biometrics using the lognormality principle, in: R. Plamondon, A. Marcelli, M.A. Ferrer (Eds.), The Lognormality Principle and its Applications, World Scientific, 2020.
[33] J.C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proenca, J. Fierrez, GANprintR: improved fakes and evaluation of the state of the art in face manipulation detection, IEEE J Sel Top Signal Process 14 (5) (2020) 1038–1048.
[34] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, 2014, pp. 2672–2680. Cambridge, MA, USA
[35] J. Yoon, D. Jarrett, M. Van der Schaar, Time-series generative adversarial networks, in: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019, pp. 5508–5518.
[36] C. Shen, Z. Cai, X. Guan, R. Maxion, Performance evaluation of anomaly-detection algorithms for mouse dynamics, Computers & Security 45 (2014) 156–171.
[37] R. Tolosana, P. Delgado-Santos, A. Perez-Uribe, R. Vera-Rodriguez, J. Fierrez, A. Morales, Deepwritesyn: On-line handwriting synthesis via deep short-term representations, in: AAAI Conf. on Artificial Intelligence (AAAI), 2021.
[38] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, J. Gonzalez-Rodriguez, Adapted user-dependent multimodal biometric authentication exploiting general information, Pattern Recognit Lett 26 (16) (2005) 2628–2639.

Alejandro Acien received the M.Sc. in Electrical Engineering in 2015 from Universidad Autonoma de Madrid. In October 2016, he joined the Biometric Recognition Group - ATVS at the Universidad Autonoma de Madrid, where he is currently collaborating as an assistant researcher pursuing the Ph.D. degree. The research activities he is currently working in Behaviour Biometrics, Human-Machine Interaction, Cognitive Biometric Authentication, Machine Learning and Deep Learning.



Aythami Morales received his M.Sc. degree in Telecommunication Engineering in 2006 from ULPGC. He received his Ph.D degree from ULPGC in 2011. He performs his research works in the BiDA Lab at Universidad Autónoma de Madrid, where he is currently an Associate Professor. He has performed research stays at the Biometric Research Laboratory at Michigan State University, the Biometric Research Center at Hong Kong Polytechnic University, the Biometric System Laboratory at University of Bologna and Schepens Eye Research Institute. His research interests include pattern recognition, machine learning, trustworthy AI, and biometrics. He is author of more than 100 scientific articles published in international journals and conferences. He has received awards from ULPGC, La Caja de Canarias, SPEGC, and COIT. He has participated in several National and European projects in collaboration with other universities and private entities such as ULPGC, UPM, EUPMt, Accenture, Unión Fenosa, Soluziona, BBVA.

Ruben Vera-Rodriguez received the M.Sc. degree in telecommunications engineering from Universidad de Sevilla, Spain, in 2006, and the Ph.D. degree in electrical and electronic engineering from Swansea University, U.K., in 2010. Since 2010, he has been affiliated with the Biometric Recognition Group, Universidad Autonoma de Madrid, Spain, where he is currently an Associate Professor since 2018. His research interests include signal and image processing, pattern recognition, HCI, and biometrics, with emphasis on signature, face, gait verification and forensic applications of biometrics. Ruben has published over 100 scientific articles published in international journals and conferences. He is actively involved in several National and European projects focused on biometrics. Ruben has been Program Chair for the IEEE 51st International Carnahan Conference on Security and Technology (ICCST) in 2017; the 23rd Iberoamerican Congress on Pattern Recognition (CIARP 2018) in 2018; and the International Conference on Biometric Engineering and Applications (ICBEA 2019) in 2019.

Julian Fierrez received the M.Sc. and the Ph.D. degrees from Universidad Politecnica de Madrid, Spain, in 2001 and 2006, respectively. Since 2004 he is at Universidad Autonoma de Madrid, where he is Associate Professor since 2010. His research is on signal and image processing, AI fundamentals and applications, HCI, forensics, and biometrics for security and human behavior analysis. He is Associate Editor for Information Fusion, IEEE Trans. on Information Forensics and Security, and IEEE Trans. on Image Processing. He has received best papers awards at AVBPA, ICB, IJCB, ICPR, ICPRS, and Pattern Recognition Letters; and several research distinctions, including: EBF European Biometric Industry Award 2006, EURASIP Best PhD Award 2012, Miguel Catalan Award to the Best Researcher under 40 in the Community of Madrid in the general area of Science and Technology, and the IAPR Young Biometrics Investigator Award 2017. Since 2020 he is member of the ELLIS Society.